Reporte de Investigación

Comunicación Bluetooth para Sensores utilizados en Aplicaciones de Control de Tráfico

M. en C. Raymundo Barrales Guadarrama
M. en C. Melitón Ezequiel Rodríguez Rodríguez
Dr. Víctor Rogelio Barrales Guadarrama

Universidad Autónoma Metropolitana-Azcapotzalco
División de Ciencias Básicas e Ingeniería
Departamento de Electrónica
Grupo de Sensores y Señales

Dr. Antonio Mocholí Salcedo
Ing. Alexander Arroyo Núñez

Universidad Politécnica de Valencia
Departamento de Ingeniería Electrónica
Grupo de Sistemas de Control de Tráfico

UNIVERSIDAD AUTONOMA METROPOLITANA
Casa abierta al tiempo
Azcapotzalco

REPORTE DE INVESTIGACIÓN

COMUNICACIÓN BLUETOOTH PARA SENSORES UTILIZADOS EN APLICACIONES DE CONTROL DE TRÁFICO

UNIVERSIDAD
AUTONOMA
METROPOLITANA
Casa abierta al tiempo Azcapotzalco

REPORTE DE INVESTIGACIÓN

COMUNICACIÓN BLUETOOTH PARA SENSORES UTILIZADOS EN APLICACIONES DE CONTROL DE TRÁFICO

M. en C. Raymundo Barrales Guadarrama
M. en C. Melitón Ezequiel Rodríguez Rodríguez
Dr. Víctor Rogelio Barrales Guadarrama

Universidad Autónoma Metropolitana-Azcapotzalco
División de Ciencias Básicas e Ingeniería
Departamento de Electrónica
Grupo de Sensores y Señales

Dr. Antonio Mocholí Salcedo
Ing. Alexander Arroyo Núñez

Universidad Politécnica de Valencia
Departamento de Ingeniería Electrónica
Grupo de Sistemas de Control de Tráfico

UNIVERSIDAD AUTONOMA METROPOLITANA
Casa abierta al tiempo Azcapotzalco

UNIVERSIDAD AUTONOMA METROPOLITANA

Casa abierta al tiempo **Azcapotzalco**

UNIVERSIDAD AUTÓNOMA METROPOLITANA

Dr. Enrique Pablo Alfonso Fernández Fassnacht
RECTOR GENERAL

Mtra. Iris Edith Santacruz Fabila
SECRETARIA GENERAL

UNIDAD AZCAPOTZALCO

Mtra. Gabriela Paloma Ibáñez Villalobos
RECTÓRA

Ing. Darío Eduardo Guaycochea Guglielmi
SECRETARIO ACADÉMICO

DIVISIÓN DE CIENCIAS BÁSICAS E INGENIERÍA

Dr. Emilio Sordo Zabay
DIRECTOR

Dr. Gabriel Soto Cortés
SECRETARIO ACADÉMICO

Dr. Andrés Ferreyra Ramírez
JEFE DEL DEPARTAMENTO DE ELECTRÓNICA

C.P. Rosa Ma. Benítez Mendoza
JEFA DE LA OFICINA DE PRODUCCIÓN EDITORIAL Y DIFUSIÓN

Comunicación bluetooth para sensores utilizados en aplicaciones de control de tráfico.
Primera edición, 2011

Universidad Autónoma Metropolitana-Unidad Azcapotzalco
Av. San Pablo 180, Col. Reynosa Tamaulipas
Del. Azcapotzalco, C. P. 02200, México, DF

Producción Editorial. Oficina de Producción Editorial y Difusión de la DCBI-A
Diseño gráfico D.C.G. Juan Manuel Galindo Medina

ISBN del libro: 978-607-477-430-6

Impreso en México/ Printed in Mexico

TABLA DE CONTENIDOS

1. Antecedentes

Las actividades humanas cuya finalidad es la de obtener bienes convenientes de consumo o mantener el bienestar de las personas (i.e. la producción de bienes, la generación de energía, la medicina, el trasporte, etc.) requieren de un entorno controlado que permita obtener bienes y servicios de las características deseadas. Es posible controlar el entorno de estas actividades analizando la información que éstas generan y, en base a la extracción de parámetros de información útiles, modificando la forma en la cual se desarrollan. El transporte de personas y bienes es una actividad que no escapa de ser controlada con el fin de rea-lizarla sin accidentes, con rapidez, con eficiencia y sin deteriorar el medio ambiente. Estas son características deseables de un buen sistema de transporte.

Como en otro tipo de actividades humanas, el control del transporte se realiza mediante dispositivos que generan señales útiles para alcanzar las características mencionadas. Actualmente, los Sistemas de Transporte Inteligente (Intelligent Transport Systems-ITS) utilizan diversos tipos de sensores para generar las señales útiles para mantener un buen sistema de trasporte ya que, debido al aumento poblacional, el incremento de las urbanizaciones y el aumento del número de automóviles, el tráfico de vehículos provoca atascos, accidentes y retardos que degradan las buenas características del transporte.

El control del tráfico vehicular se vale de obtener imágenes y patrones, monitorear el medio ambiente e interactuar con infraestructuras para ayudar a mantener las características del sistema de transporte en una ciudad o en vías de comunicación terrestre. Puesto que el vehículo se encuentra en movimiento cuando opera dentro del sistema de transporte, las señales que el vehículo requiere para lograr cambios automáticos en su velocidad y su ruta, advertir de su presencia, de su proximidad y recibir mensajes de alerta deben enviarse y/o recibirse sin utilizar un medio físico de comunicación. Por lo tanto, los medios de comunicación inalámbricos cobran especial importancia en los ITS.

Aunque existen aplicaciones ITS particulares que requieren de sistemas de comunicación de largo alcance (GPS, telefonía, localización por radar, etc.), muchas de las aplicaciones ITS para el control del tráfico utilizan sistemas de comunicación de corto alcance. Aunque en los años 80s del s. XX se llevaron a cabo iniciativas para establecer un ITS normalizado (el proyecto japonés JSK (Association of Electronic Technology for Auto-mobile Traffic and Driving), el proyecto PATH en California, el proyecto Chauffer en la UE, etc. [1] donde se diseñaron sistemas de comunicación de corto alcance para el control del tráfico vehicular, es hasta el año 2001 que se inicia un serio y ambicioso proyecto ITS dirigido en los Estados Unidos de América que, en principio, se aplicaría también en Canadá y México y que ha establecido normas para definir sistemas dedicados de corto alcance (Dedicated Short Range Communications—DSRC) para el control del tráfico de vehículos. Estos sistemas operarían en un rango de 1 000 m de distancia, trasmitirían su información en la banda comprendida en el intervalo de los 5.850 a los 5.925 GHz, alcanzarían razones de transmisión de 27 Mb/s y el intercambio de información se realizaría con medios basados en la norma IEEE 802.11p [2]. Estos sistemas intercambiarían entonces grandes cantidades de información (imágenes, video, internet, etc.). Sin embargo, no se espera que las normas de comunicación correspondientes estén listas antes del año 2009 [3].

Sin embargo, la información manejada por algunos dispositivos para el control del tráfico no requiere comunicarse mediante dispositivos capaces de recibir o transmitir grandes cantidades de datos. Es el caso, de los sistemas de cobro de cuotas, el control y monitoreo de infraestructuras, la vigilancia de las condiciones meteorológicas o información intrínsecas al vehículo. El ITS norteamericano vislumbra que estándares de co-

municación como Bluetooth o Zigbee se utilicen exclusivamente para la comunicación intrínseca de las señales del vehículo [4]. Sin embargo, el éxito comercial de Bluetooth o Zigbee y sus recientes avances en ahorro de potencia, alcance de la comunicación y velocidad, los convierten en medios inalámbricos de comunicación que pueden ofrecer soluciones ITS mientras los estándares finales se completan e implementan.

Al presente, Zigbee ofrece la plataforma más conveniente para el manejo de información generada por sensores debido a su rapidez de respuesta (15 ms), bajo consumo (años de servicio de la batería) y posibilidad de interconectar hasta 65 000 dispositivos en red [5]. Estas características son importantes cuando se quiere tener conexiones punto a punto entre un sensor y su monitor o una red de sensores porque, en general, los sensores se han vuelto "inteligentes" ("smart sensor" [6]), es decir, deben de ser capaces de adquirir, procesar y transmitir la información de la señal que manejan y, posiblemente, servir como "routers" en redes de sensores Ad Hoc [7]. Las aplicaciones ITS donde se requieran sensores en áreas abiertas o donde la rapidez de despacho de una señal de advertencia sea importante, podrán beneficiarse de las bondades de los sistemas Zigbee de comunicación.

Bluetooth es el nombre corriente de un sistema de comunicación de datos cuyas principales características (alcance máximo 100 m, transferencia máxima 1 Mbps, rapidez de respuesta 3 s, 7 esclavos por piconet) [5] lo han hecho muy útil en el reemplazo de cables que comunican datos entre una PC y sus periféricos. Aunque su rapidez de respuesta (el tiempo para realizar una conexión inalámbrica entre dos dispositivos) y su nivel de consumo (agotaría una batería de 3 V en menos de 3 días), comparado con Zigbee, no lo han hecho un sistema de comunicación preferido para implementar redes de sensores [5]. Su popularidad, costo competitivo, mejoras continuas y su uso en ambientes industriales [8], lo hacen atractivo para aplicaciones ITS de corto alcance donde:

- no se requiera autonomía del dispositivo,
- se toleren tiempos de conexión moderadamente lentos (< 3 s) y
- no se requiera una red de dispositivos extensa (no más de 3 esclavos y un maestro).

Aplicaciones ITS específicas con las características antes mencionadas son:

- sistemas de aproximación de vehículos lentos
- sistemas de extracción de características vehiculares de bajo volumen de información (imágenes fijas preprocesadas, señales de espiras en las vías, vigilancia de las condiciones del clima, control de infraestructuras, etc.)
- duplicación del entorno de señalización de las vías en el tablero del automóvil, etc.

Esta memoria técnica tiene como propósitos:

- reportar la implementación de una plataforma de desarrollo para ayudar a integrar la capacidad de comunicación Bluetooth a sensores orientados a aplicaciones ITS con salida RS-232
- reportar la utilidad de la plataforma en la puesta en marcha de un sensor electromagnético para la extracción de las características de un vehículo
- reportar las principales características del sensor y su medio de transmisión.

2. JUSTIFICACIÓN DEL USO DE LA COMUNICACIÓN INALÁMBRICA EN LOS SENSORES ITS

Los sensores inteligentes, en general, presentan la estructura mostrada en la Fig. 1 [6].

Durante la fase de desarrollo de un sensor inteligente, es común que el diseñador utilice herramientas de desarrollo especializadas que incluyan algunos de los bloques del diagrama de la Fig. 1. De esta manera, es posible diseñar sensores de diversos usos, de acuerdo a las variables de entrada que tienen que sensar. Puesto que el sensor, el amplificador y el procesamiento analógico dependen de las características de la señal a medir y de la aplicación, se opta por diseñar un acondicionador analógico de propósito particular e interconectarlo a la herramienta de desarrollo de propósito general. Generalmente, los bloques que una herramienta de desarrollo incluye son:

- la conversión de datos,
- el procesamiento digital,
- el procesador de control y
- la comunicación de datos.

Cuando la arquitectura del sensor inteligente se ha probado, se puede fabricar un circuito impreso que contenga la solución material final.

Se puede observar que la comunicación de datos es inherente a la arquitectura del sensor inteligente y, por lo tanto, constituye un puerto físico de salida de la señal procesada del sensor que se tendrá que comunicar a su destino final: un acumulador de datos, un monitor de datos, un controlador, un medidor, etc. El medio para comunicar esta salida hacia el destino final puede ser mediante un bus físico o un medio inalámbrico. Por lo tanto, es menester incorporar al ciclo de pruebas el medio de comunicación. Si el medio es inalámbrico, es conveniente implementar una plataforma de desarrollo que facilite las pruebas de transmisión de la información del sensor.

En muchos casos, la salida final del sensor inteligente la constituye el puerto de salida del procesador de control. Este puerto, en general, es un puerto serie y puede ser del tipo USB, RS-232 o SPI. Por lo tanto, la plataforma de desarrollo debe contemplar una entrada de datos formada por un puerto serie compatible con la salida del sensor inteligente.

La plataforma de desarrollo que se describe en esta memoria, ha tenido en cuenta los recursos disponibles y la salida de los sensores desarrollados en el laboratorio del Grupo de Sistemas de Control de Tráfico del ITACA en la UPV. Las consideraciones que han determinado el diseño de la plataforma fueron:

- la salida de los sensores desarrollados en el laboratorio (espiras, sensores de lluvia, de temperatura, etc.) comunican su salida a través de un puerto RS-232
- los sensores sólo se utilizarían en comunicaciones punto a punto (sensor a display, a monitor o a infraestructura), no se diseñarían para ser

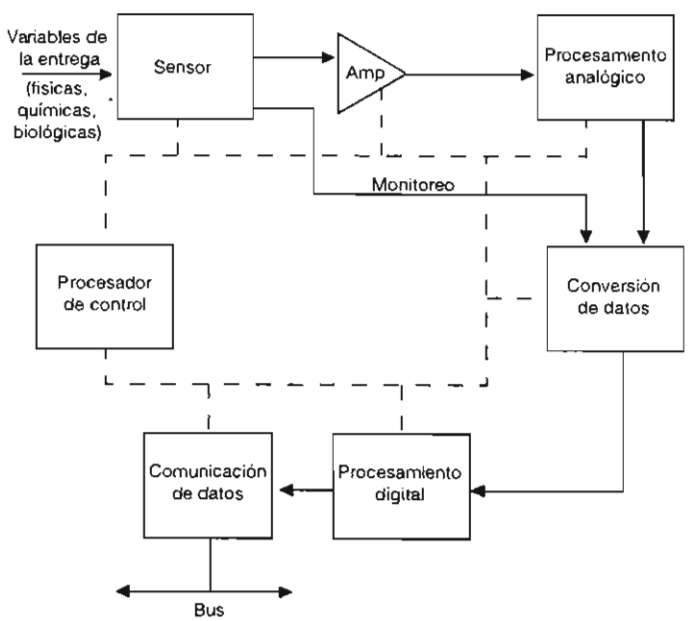


Fig. 1. Diagrama a bloques de un sensor inteligente. Tomado y adaptado de [6].

autónomos (alimentación local continua o acumulador del automóvil) y las variables de entrada no generan un volumen alto de información (ancho de banda grado instrumentación)

- la tecnología de comunicación inalámbrica más conveniente para el tipo de sensores desarrollados y con los recursos disponibles para implementarla, fue Bluetooth y
- se contaba con instrumentos virtuales para la presentación de los resultados de las pruebas de transmisión y/o recepción.

Puesto que el enlace Bluetooth tenía que responder al modo particular de recepción de la información, se requería contar con una herramienta que permitiera la configuración del enlace Bluetooth. En este caso, se utilizó la herramienta CASIRA de CSR [9] para programar un módulo de comunicación con la pila ("stack") de las capas del protocolo Bluetooth incorporado en el módulo. Esta herramienta es de uso corriente en otros laboratorios y era el recurso disponible. Ha sido una herramienta conveniente porque incluye puertos de E/S, entre ellos, un puerto RS-232. El dispositivo Bluetooth que recibe los datos del sensor es uno de tipo comercial ("dongle") que puede conectarse a un puerto USB de la PC y cuenta con software de configuración propio.

3. Implementación de la plataforma

La Fig. 2 muestra el diagrama a bloques de la plataforma de desarrollo para sensores ITS con capacidad Bluetooth.

3.1 Descripción de la plataforma

Transmisión bajo el protocolo Bluetooth

Del lado izquierdo de la Fig. 2, se esquematizan los componentes que permiten la transmisión de los datos del sensor.

La herramienta CASIRA es un sistema que incorpora programación y soluciones materiales[1] para realizar las funciones del protocolo de comunicaciones Bluetooth [10]. Esto permite a un diseñador integrar las bondades de Bluetooth a su aplicación. En este caso, se trata de integrar capacidad Bluetooth a un sensor. El módulo cumple con la especificación 1.1 (radio clase 2 de 10 m) y un consumo aproximado de 874 mW.

El puerto RS-232 está integrado a la herramienta de desarrollo CASIRA y ésta permite la programación, desde una PC y utilizando lenguaje ANSI C, de un módulo que contiene todas las capas necesarias para construir el "stack" de las capas del protocolo de comunicación Bluetooth. La Fig. 3 muestra un diagrama a bloques de la herramienta.

Una foto de la herramienta CASIRA, donde se indican sus partes interesantes, se muestra en la Fig. 4.

Las características de la herramienta CASIRA que han permitido el desarrollo del sistema de transmisión del sensor son [11]:

a) Las capas del protocolo completo Bluetooth embebidas en un chip BlueCore2-External. Este protocolo es configurable de acuerdo a la repartición del protocolo entre un controlador ("host") y el chip que contiene las capas inferiores del protocolo. La Fig. 5 muestra un diagrama a bloques del sistema del chip.

b) Un módulo BC2MOD2D, compatible con la versión 1.1 de la especificación, que contiene los

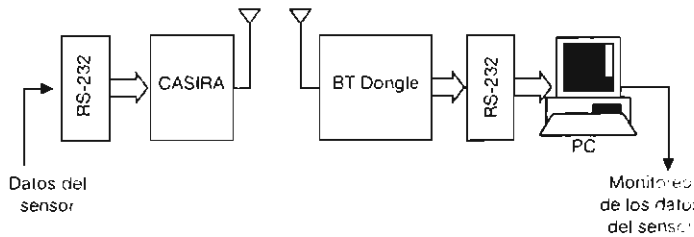


Fig. 2. Diagrama a bloques de la plataforma de desarrollo para sensores ITS con capacidad Bluetooth.

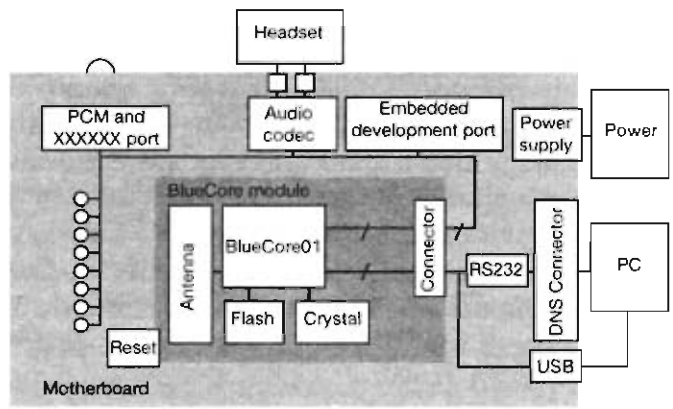


Fig. 3. Diagrama a bloques de la herramienta de desarrollo CASIRA [10].

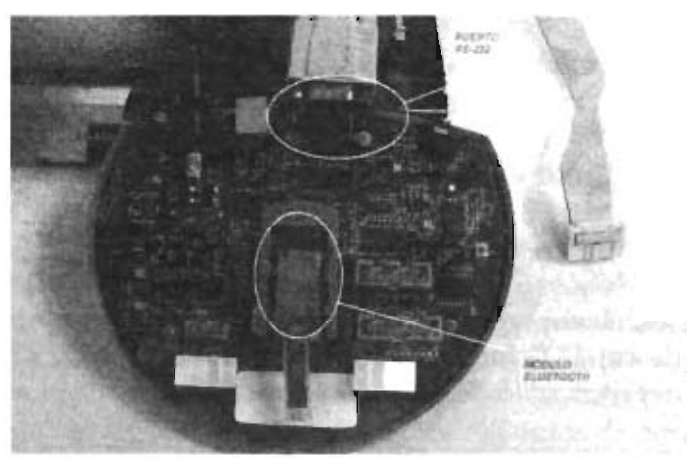


Fig. 4. Fotografía de la herramienta CASIRA.

1 El término utilizado en el medio es "embedded system".

componentes básicos para implementar, parcial o completamente, el protocolo Bluetooth y permitir el enlace mediante un radio. Está compuesto[2] principalmente de un chip BlueCore2-External, una antena optimizada para la transmisión/recepción de RF, memoria flash para la grabación de la programación del usuario, un cristal para el reloj de referencia y un bus de comunicaciones. La Fig. 6 muestra una foto del módulo.

c) Un puerto RS-232.

Recepción bajo el protocolo Bluetooth

Del lado derecho de la Fig. 2, se esquematizan los componentes que permiten la recepción de los datos del sensor remoto.

El "BT dongle" es un dispositivo de comunicación Bluetooth comercial cuya única función será la de recibir los datos del sensor remoto enviados bajo el protocolo Bluetooth. El dispositivo se conecta vía USB a una PC y el software asociado se instala fácilmente en ambientes de Windows XP. El dispositivo cumple con la versión 1.1 de la especificación Bluetooth, tiene un alcance de 30m, una transferencia de 1 Mbps y un consumo de 330 mW. La Fig. 6 muestra el aspecto de este dispositivo BT ver. 1.1

La visualización de los datos recibidos se ha realizado gracias a una Hyperterminal de Windows que se configura para conectarse al puerto serie virtual creado por el dispositivo Bluetooth.

Probablemente, el lector se pregunte por qué se ha escogido realizar la comunicación física de los datos mediante puertos RS-232. En la sección 5 se explica la razón de ello.

3.2 Breve descripción de la tecnología Bluetooth empleada en el CASIRA

La tecnología Bluetooth

La tecnología Bluetooth se organiza en una pila[3] de capas como la que define el modelo ISO/OSI del protocolo de comunicaciones (Fig. 7).

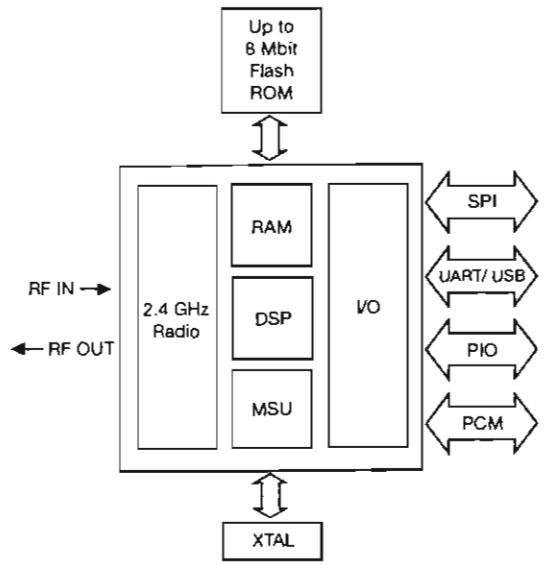


Fig. 5. Diagrama a bloques del chip BlueCore2- External [11].

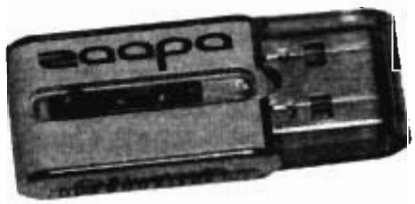


Fig. 6. Fotografía del dispositivo Bluetooth ZBTA-3130 de la casa Zaapa.

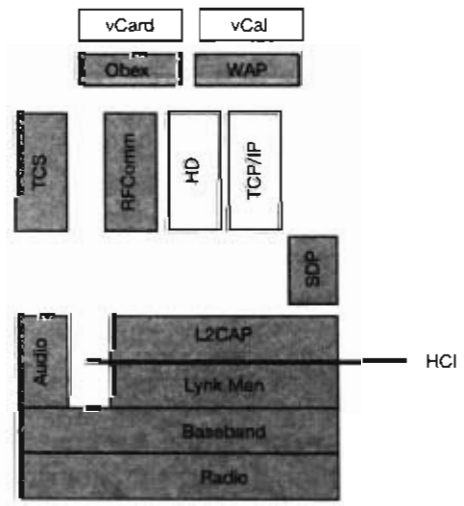


Fig. 7. Una ilustración de la pila del protocolo Bluetooth. Tomado y adaptado de [12].

2 Ver Fig. 3.

3 Conocida en el medio como "stack".

Las frecuencias de transmisión se generan en la capa física llamada "Radio" y los datos se modulan para coincidir con el rango de frecuencias de la banda ISM[4]. La capa "Baseband" controla los saltos de frecuencia y es responsable de los errores en la conexión y del establecimiento de enlaces sincrónicos (SCO) y asincrónicos (ACL). El "Link Manager Protocol (LMP)" controla el status de los enlaces existentes y de los modos de ahorro de energía y proporciona mecanismos de seguridad como la autenticación y la codificación. El "Host Controller Interfaces (HCI)" garantiza inter-operabilidad mediante la definición de interfases normalizadas entre las capas superiores y las inferiores. Se le puede utilizar para separar la pila del protocolo (ver sección siguiente) y conectar dos secciones de la pila vía interfases físicas tales como un USB o un UART. El Control Lógico del Enlace y el Protocolo de Adaptación ("Logical Link Control and Adaptation Protocol" (L2CAP)) crea la transición entre las conexiones lógicas y físicas mediante la gestión del multiplexado y la segmentación de los datos. RFCOMM es una capa superior del protocolo que emula una conexión serie sobre un enlace L2CAP, mientras que la Especificación del Protocolo para el Control de Telefonía ("Telephony Control Protocol Specification" (TCS)) permite voz sobre IP[5] y el Protocolo para el Descubrinmiento de Servicios ("Service Discovery Protocol" (SDP)) descubre los servicios disponibles o perfiles en los dispositivos de la vecindad.

Dependiendo de la clase del radio, se tendrán distintos alcances de la señal. La Tabla I muestra la especificación correspondiente.

Aplicaciones equipadas con radios Bluetooth (tal como un sensor inalámbrico) pueden establecer una conexión punto-a-punto (sólo dos dispositivos Bluetooth en juego). Este esquema de conexión es el que se presenta en muchas aplicaciones

Tabla I. Alcances de la señal según el tipo de radio Bluetooth.

Clase del Radio	Rango	Aplicación
3	1 m	Red de área personal
2	10 m	Dispositivos portátiles
1	100 m	Uso industrial

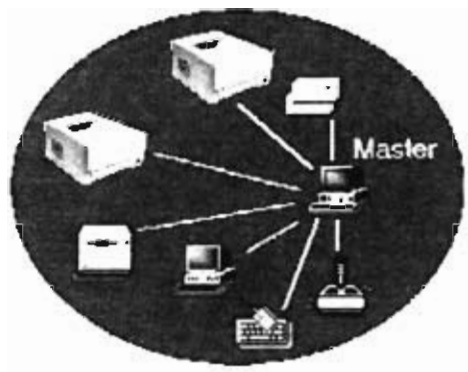


Fig. 8. Una ilustración de una piconet. Tomado y adaptado de [13].

ITS de corto alcance. Los módulos Bluetooth permiten también conexiones punto-multipunto. En este tipo de esquema, varias unidades Bluetooth comparten el canal físico y dos o más unidades que comparten el mismo canal forman una piconet (Fig. 8). En una piconet sólo se permite la existencia de un módulo Bluetooth maestro ("master"), mientras que los otros módulos actúan como esclavos ("slaves"). Este esquema puede ser útil cuando un vehículo o una infraestructura, en un ambiente ITS, ha de recibir varias señales.

La organización de la pila Bluetooth en el CASIRA

El modulo BlueCore y la tarjeta madre del CASIRA forman un sistema embebido susceptible de programarse en ANSI C. El software[6] que el fabricante proporciona junto con la herramienta es el complemento del kit de desarrollo. La colección de librerías, el compilador C, los ejemplos de apli-

4 ISM es el acrónimo de "Industrial-Scientific-Medical" que designa una banda de frecuencias en el intervalo 2.40-2.48 GHz. Los radios Bluetooth pueden transmitir en esta banda hasta 1MS/s y utilizan un patrón de comunicación por saltos de frecuencia ("frequency hopping") para evitar interferencias.

5 VoIP.

6 BlueLab software development kit (SDK).

cación y la documentación de ayuda forman un conjunto de archivos, llamado BlueLab, útiles para el desarrollo de aplicaciones finales con capacidad Bluetooth. La versión del BlueLab utilizada en este trabajo fue la 2.8 y el compilador C fue el xap-local-xap-gcc funcionando bajo ambiente Cygwin.

El BlueLab permite escoger al usuario de la herramienta entre tres diferentes maneras de hacer corresponder la pila Bluetooth en una solución material final [14] (Fig. 9).

Estas son formas de operación del chip BlueCore 2-External y se conocen como modelos. Los modelos son:

a) Pila BlueCore HCI (BlueCore HCI Stack). En este modelo, la pila del protocolo se separa por encima de la capa del gestor de enlace ("link manager layer—LM"); las capas por arriba del LM residen en un sistema anfitrión ("host") y las capas por abajo del LM en el módulo. El HCI define la comunicación entre estas dos secciones. Este es el esquema de trabajo cuando se utilizan sistemas basados en PCs o estaciones de trabajo, donde se cuenta con suficiente poder de cómputo para manejar las operaciones de las capas superiores de la pila. Este modelo permite razones de transmisión máximas de 723 Kbps (unidireccional) y soporta 7 esclavos por piconet. Este modelo es muy flexible.

b) Pila BlueCore RFCOMM (BlueCore RFCOMM Stack). La pila del protocolo se separa por encima de la capa RFCOMM y esa parte se hace residir en un sistema anfitrión (host). Esta implementación se utiliza cuando se cuenta con un microcontrolador externo con suficientes recursos para dar servicio a la capa de aplicación. En este modelo, la razón de transmisión/recepción baja a 350 Kbps y una piconet consistiría solamente de 3 esclavos y 1 maestro debido a las limitaciones físicas del módulo. En este modelo, se reduce el hardware y el software asociados al "host" a costa de sacrificar la potencia y flexibilidad del modelo anterior.

c) Pila BlueCore con Máquina Virtual (BlueCore Virtual Machine Stack). Este modelo es interesante porque opera como un sistema embebido desde el punto de vista que no necesita hardware o software adicional funcionando en diferentes sitios de la aplicación total. La pila completa y la aplicación del usuario residen en el módulo. Se utiliza una máquina virtual ("virtual machine"—VM) que funciona en el microcontrolador interno del módulo y que se utiliza para ejecutar la aplicación en el mismo chip. Un inconveniente de este modelo es la reducción en el rendimiento debida a la potencia de cómputo adicional requerida para soportar una pila más grande que en los modelos anteriores. Esto se refleja en una reducción de la velocidad de transmisión por debajo de los 200 Kbps (se han reportado proporciones aún más bajas del or-

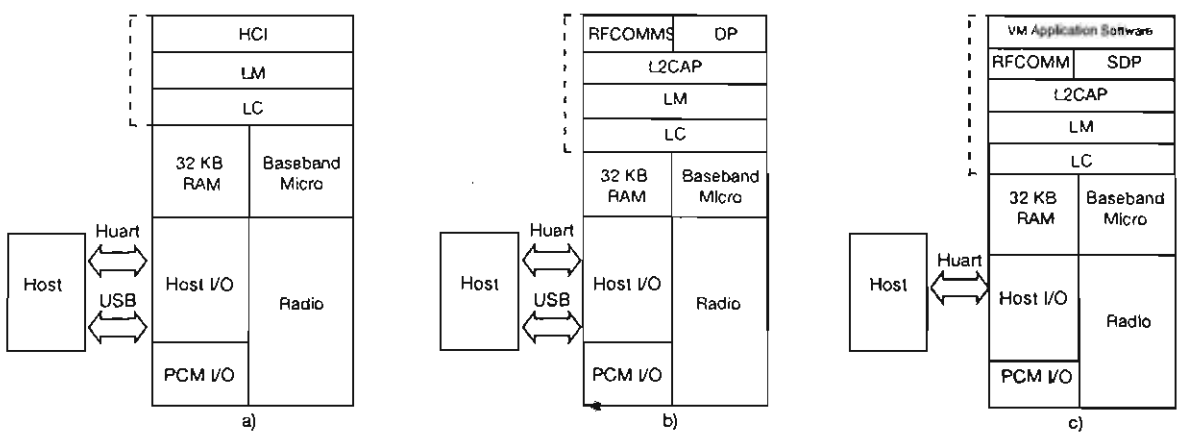


Fig. 9. Los tres modelos operativos del modulo BlueCore 2-External: a) Pila BlueCore HCI; b) Pila Bluecore RFCOMM; c) Pila BlueCore Virtual Machine [14].

den de 3 Kbps). Sin embargo, es posible tener hasta 3 esclavos simultáneamente conectados por cada maestro.

Puesto que se ha pensado en proporcionar máximo soporte a desarrolladores de aplicaciones ITS, se ha elegido el tercer modelo de operación de la pila Bluetooth del módulo BC2MOD2D asociado al CASIRA. De esta manera, el desarrollador no necesitaría de implementar hardware y software adicional a la estructura del sensor ITS y la tarea de comunicar la aplicación con el módulo Bluetooth se completaría en poco tiempo. El inconveniente del modelo de la máquina virtual (i.e. la velocidad de la transferencia de datos) es aceptable para enlaces punto a punto donde se intercambia información a bajas tasas de datos (<200 kbps), que es el caso en muchas aplicaciones ITS.

Puertos serie, UARTs y la capa RFCOMM

En general, las líneas para la recepción y la transmisión de datos de un puerto serie se conectan a una UART (Universal Asynchronous Receiver Transmitter). El trabajo de la UART es convertir los datos serie trasportados por los cables del puerto a datos en paralelo utilizados por el dispositivo de procesamiento. Las UARTs utilizan "buffers" para convertir entre datos en serie y en paralelo. Esto permite reducir la carga del procesador. En lugar de que el procesador sea interrumpido para leer cada bit enviado por los cables, la UART transfiere los bits entre los cables y el buffer y luego el procesador sólo interviene cuando el buffer está completamente lleno.

La UART es un periférico del sistema, así que sus líneas de direccionamiento aparecen en el mapa de direcciones del sistema. Algunos procesadores reservan un rango especial de direcciones para la entrada/salida. Otros sistemas las mapean a cualquier parte de la memoria normal. Puesto que las UARTs lucen como áreas de memoria para un microprocesador, es posible emular un puerto serie en software tomando un área de memoria y colocando valores en ella como lo haría una UART real.

La capa RFCOMM, de la pila Bluetooth, emula los nueve circuitos de un puerto serie RS-232 y, también, especifica cómo se puede emular una riada ("stream") de datos serie. Puesto que la riada de datos serie de un puerto RS-232 es vista por el microprocesador como habiendo sido enviada a través de una UART, el software asociado a los puertos serie emulados manejan los datos en paralelo. De manera similar, el software de la capa RFCOMM trata a los datos, enviados por las capas inferiores de la pila Bluetooth, en paralelo.

Una UART se conecta físicamente en el sistema mediante alambres y buffers. La capa RFCOMM se conecta, por software, a las capas inferiores de la pila vía la capa L2CAP [15].

Tipos de dispositivos RFCOMM

La capa RFCOMM da servicio a dos tipos de dispositivos:

- Tipo 1-Puerto serie internamente emulado (o su equivalente).
- Tipo 2-Dispositivo intermedio con puerto serie físico.

En la Fig. 10, se muestra la pila del protocolo para el RFCOMM tipo 1.

La entidad que emula el puerto hace corres-ponder un sistema de interfase de comunicación específico (API) a los servicios RFCOMM. Esto se utiliza para conectar aplicaciones ya existentes, como

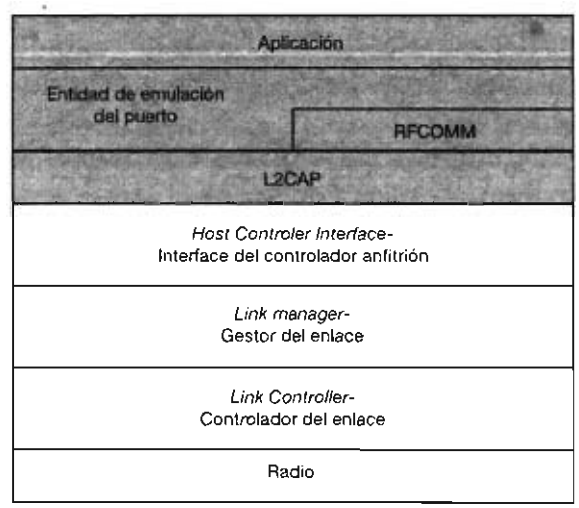


Fig. 10. Ilustración conceptual del dispositivo RFCOMM tipo 1. Tomado y adaptado de [15].

lo muestra la Fig. 10, o bien conectar aplicaciones específicamente escritas para Bluetooth. Habitualmente, un dispositivo tipo 1 es el final de un camino de comunicación (i.e. una PC o una impresora).

En la Fig. 11, se muestra la pila del protocolo para el RFCOMM tipo 2.

La entidad de puerto proxy pasa datos de la capa RFCOMM a una interfase RS-232 externa enlazada a otro dispositivo. Los dispositivos RFCOMM del tipo 2 son dispositivos intermedios que se colocan a la mitad del trayecto de la comunicación. Un modem o un sensor son ejemplos de dispositivos del tipo2.

Puesto que se trata de desarrollar una plataforma de comunicaciones Bluetooth para sensores con salida RS-232 y esta salida es física, el programa residente en la memoria flash del módulo deberá utilizar funciones o librerías que le permitan interactuar con el dispositivo externo (Data Communications Equipment—DCE). Afortunadamente, el SDK del CASIRA cuanta con tales librerías. Por lo tanto, la plataforma desarrollada constituye un dispositivo RFCOMM del tipo 2.

3.3 Preparación de la conexión del módulo Bluetooth

Antes de iniciar la comunicación de datos Bluetooth punto a punto, es necesario definir:

- el papel de cada punto (i.e. esclavo o maestro),
- el perfil de la aplicación útil para los propósitos del usuario,
- el juego de servicios mínimo requerido por la aplicación y
- la configuración correcta acorde con la aplicación.

Algunas de las definiciones anteriores requieren cierta familiarización con el modo de funcionamiento de la pila Bluetooth. La descripción de este funcionamiento es extensa y no es el propósito de esta memoria presentarla. Las referencias [15], [16], ayudarán a lectores, quienes se inician en el uso de enlaces Bluetooth, a comprender detalles que se obvian en las siguientes secciones. Sin embargo, algunas notas descriptivas se añaden cuando se considera pertinente.

Las definiciones antes mencionadas se especifican en la Tabla II.

Tabla II. Definiciones para la pila Bluetooth de la aplicación presente

Requerimiento	Especificación
Papel de cada punto	El sensor funcionará como esclavo. El dongle BT funcionará como maestro (iniciará la solicitud de recepción de datos).
Perfil de la aplicación	Perfil de puerto serie*
Juego de servicios mínimo	- Servicio de definición del rol (maestro o esclavo) - Servicio de búsqueda ("inquiring")
Configuración necesaria	Dirección UID del puerto serie Parámetros de operación del puerto RS-232 Habilitación de la UART

**Los perfiles Bluetooth aseguran la interoperabilidad. entre diferentes dispositivos. proporcionando un juego bien definido de procedimientos en las capas superiores y formas uniformes de utilizar las capas inferiores de la pila Bluetooth. De esta manera, los perfiles ayudan a los módulos Bluetooth a conectarse con diferentes dispositivos y aplicaciones y trabajar de una forma estándar [15, p.363].*

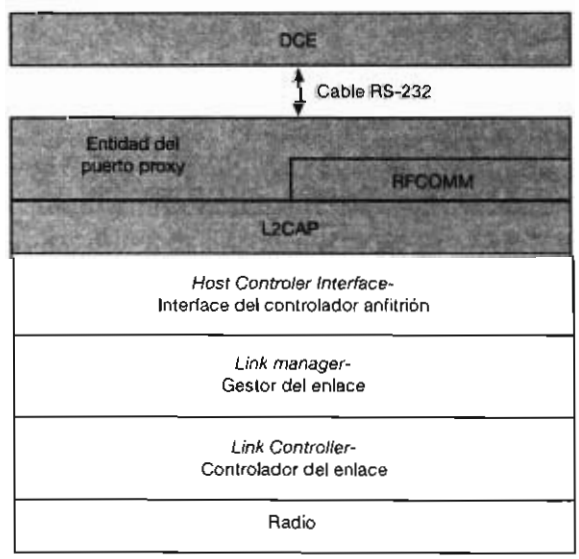


Fig. 11. Ilustración conceptual del dispositivo RFCOMM tipo 2. Tomado y adaptado de [15].

Organización del software de desarrollo

BlueLab permite escribir código para programar los módulos BlueCore de CSR. El kit CASIRA permite adquirir y grabar en el módulo la programación del usuario. Se siguieron los pasos recomendados en la documentación del fabricante [17] para instalar el paquete BlueLab y conectar y probar el kit CASIRA.

Una vez que las herramientas de programación han sido instaladas y probadas, es entonces posible relacionar una aplicación con la pila Bluetooth embebida en el módulo BlueCore. La Fig. 12 ilustra la pila formada en la memoria flash del módulo BlueCore cuando se ha elegido de trabajar con el modelo de la pila BlueCore con máquina virtual. En esa figura, nótese que las capas SDP ("Service Discovery Protocol"), L2CAP y RFCOMM están implementadas en software. Estas capas se necesitan para dar servicio al perfil de puerto serie (también se les utiliza para dar servicio a perfiles simples en el perfil de puerto serie tal como el perfil de auriculares de manos libres ("Headset profile"). Estas capas han sido escritas por la casa Mezoe y se conocen, en conjunto, como el "BlueStack". Por encima de las capas del BlueStack, un Gestor de la Conexión ("Connection Manager") maneja las conexiones a la capa RFCOMM. BlueLab proporciona la librería del Connection Manager para facilitar las conexiones aunque no es esencial utilizarla. El usuario puede escribir su propia librería del Conection Manager de acuerdo a sus necesidades.

En la parte superior de la pila se encuentra la máquina virtual. La máquina virtual permite al Connection Manager, a las librerías y al software de aplicación correr en un espacio protegido de la memoria. El software de aplicación se compila para hacerlos corresponder con los códigos de operación de la máquina virtual. Mientras la aplicación corre, la máquina virtual verifica si cada instrucción trata de realizar un acceso inválido a memoria. De esta manera, la máquina virtual garantiza que el software de aplicación no interfiera con el funcionamiento correcto de la pila Bluetooth.

Normalmente, el chip BlueCore viene cargado con una imagen del BlueStack y de la máquina virtual.

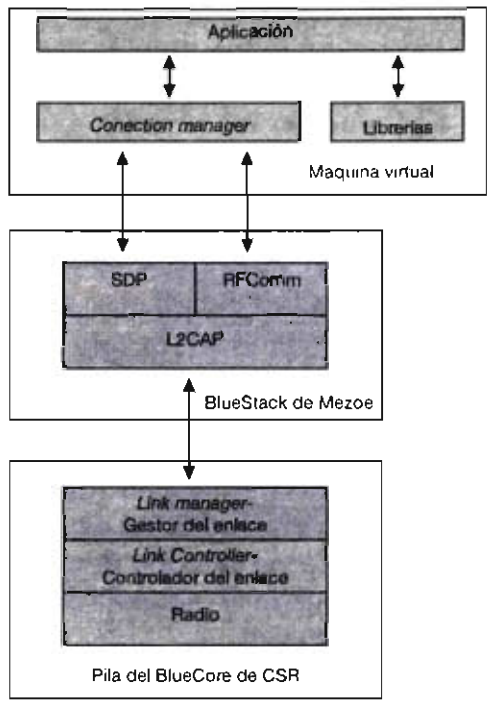


Fig. 12. Diagrama a bloques de la pila Bluetooth completa en el módulo CSR. Tomado y adaptado de [16].

El Connection Manager (CM)

El CM gestiona todas las capas de la pila del protocolo Bluetooth desde la capa RFCOMM hacia abajo. Sin el CM, se tendría la necesidad de establecer enlaces ACL[7], configurar los enlaces para RFCOMM, instalar y configurar los enlaces L2CAP y, finalmente, instalar un enlace RFCOMM. El CM realiza todas estas tareas con sólo invocarlo. El CM de BlueLab permite a la aplicación del usuario enviar paquetes sobre la capa L2CAP así como paquetes sobre la capa RFCOMM (al final, todos los paquetes enviados van sobre la capa L2CAP porque es una capa inferior a la capa RFCOMM.

Los paquetes se envían sobre una conexión y cada conexión debe conducir hacia un dispositivo de destino, por lo tanto, antes de enviar cualquier

7 Un enlace ACL ("Asynchronous Connection-Less") existe cuando se ha establecido comunicación entre un maestro y un esclavo y proporciona una conexión conmutada de paquetes donde los datos se intercambian esporádicamente siempre y cuando haya datos disponibles en la capa superior de la pila [15, p. 48].

paquete, se debe instruir al CM para establecer una conexión con un dispositivo de destino.

Los intentos para establecer la conexión, por parte del CM, pueden ser exitosos o no y, por lo tanto, alguna contingencia debe llevarse a cabo para reintentar la reconexión o indicar la razón del fallo. Durante el proceso, se tienen que determinar las características de la conexión, es decir, el juego de servicios mínimo y la configuración necesaria según las especificaciones de la Tabla II. El curso normal de "solicitudes" y "respuestas" del CM permitirán el establecimiento de una conexión segura con las características deseadas.

El proceso de conexión está constituido por varios estados y la conexión pasará a uno u otro estado de acuerdo a la verificación de ciertas condiciones. La Fig. 13 ilustra el esquema de los estados del proceso de conexión.

Los estados por los cuales pasa el proceso de conexión son:

a) Inicio (starting). En este estado se inicializa al CM. Éste no es parte de la pila del protocolo Bluetooth. Es una librería aparte y de uso opcional. Como no es una parte esencial del sistema, no se inicializa automáticamente (a diferencia de la pila del protocolo que se inicializa automáticamente para asegurar que corre apropiadamente). Por lo tanto, si se quiere utilizar el CM, es necesario inicializarlo y abrirlo enviándole ciertos mensajes.

b) Iniciar conexión (connecting). En el siguiente estado se define el rol del nodo. Es necesario enviar mensajes al CM que indiquen que el nodo se conecta como un maestro ("master") o como un esclavo ("slave"). El proceso puede permanecer en este estado mientras el rol no se haya definido.

c) Búsqueda (inquiring). La aplicación tiene que realizar una búsqueda de los dispositivos en su vecindad para poder obtener información del otro nodo. Los mensajes de búsqueda que envía la aplicación terminan, si la búsqueda ha sido exitosa, en la recepción de un paquete FHS[8] por parte del nodo remoto con información útil para el nodo buscador. Se permanece en este estado hasta que se define a qué dispositivo se hará la conexión.

d) Activo (active). En este estado se logra la comunicación y se permanece en él mientras no ocurra una condición donde el proceso tenga que abandonarlo (i.e. el nodo remoto ha abandonado la conexión, una falla momentánea de la alimentación, etc.).

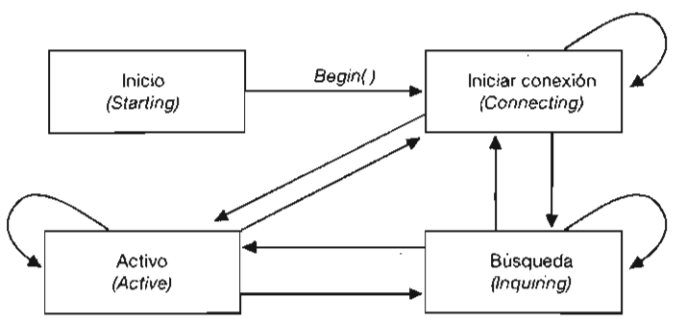


Fig. 13. Esquema de los estados del proceso de conexión de la pila BlueStack.

Los cambios de estado dependen de las condiciones en las cuales se intenta desarrollar la conexión. Por ejemplo, si la conexión se interrumpe, pero se conoce la información del nodo remoto, se pasa al estado "iniciar conexión" y se regresa directamente al estado "activo"; si, por alguna razón, se pierde la conexión y existen nuevos dispositivos en la vecindad, se hace una nueva búsqueda y se pasa al estado activo de nuevo.

Bajo el modelo de la Fig. 12, es la aplicación la que tiene que generar los mensajes al CM para terminar en el estado "activo" de la conexión. Desde el punto de vista de la programación de los sistemas embebidos, la aplicación es una "TAREA" que genera mensajes al CM el cual, a su vez, es otra "TAREA" que comunica estos mensajes convenientemente a las capas inferiores del BlueStack. El CM se trata como la Tarea 0 ("Task 0") en el entorno de la programación del sistema embebido (CASIRA + BlueCore) y es un número de tarea reservado. La aplicación también tiene un número de tarea reservado en este entorno: es la tarea 1

[8] Un paquete FHS ("Frequency Hop Synchronisation") contiene toda la información que una aplicación necesita para crear una conexión con un nodo remoto. Por ejemplo, en este paquete se recibe la información de la clase de dispositivo del nodo que se ha encontrado [15, p. 13].

("Task 1") [16]. Un esquema de la relación entre estas tareas se muestra en la Fig. 14.

Refiérase a la Fig. 14. Los mensajes de solicitud se generan desde la aplicación (Task 1) para inicializar al CM y enviar las solicitudes al BlueStack para intentar pasar a uno u otro estado del proceso de conexión. Si ha sido posible pasar a otro estado, habiendo hecho la solicitud, el CM enviará un mensaje de confirmación correspondiente. Este intercambio de información es un protocolo flexible pues el usuario lo puede definir de acuerdo a las especificaciones de su aplicación (perfil de puerto serie, perfil de telefonía, etc.) en la programación del BlueCore. En condiciones normales, y después de que el CM ha confirmado todas las solicitudes hechas por la aplicación, el dispositivo Bluetooth ha de permanecer conectado y listo para enviar información en cualquier momento.

En [16] se hace una muy buena presentación de los mensajes intercambiados entre la aplicación y el CM para llegar al estado ACTIVO de la conexión. La mayor parte de esta presentación se corresponde con los mensajes y librerías de la versión 2.82 del BlueLab Professional utilizadas en este trabajo para implementar el proceso de conexión. Sin embargo, la presentación de las librerías para enviar datos es obsoleta, pero se documentan las utilizadas en este trabajo. Se recomienda al lector referirse a [Kammer] para un entendimiento más vasto de la implementación del proceso de conexión que se realizó para esta memoria. La Fig. 15 muestra la lista de mensajes de solicitud en el orden enviado y la lista de mensajes de confirmación correspondiente. Todos los mensajes del CM están declarados en el archivo `cm_rfcomm.h`. El CM está implementado en la librería `CM_RFCOMM: libcm_rfcomm.a`.

Cuando la aplicación (Task 1) envía un mensaje al CM (Task 0), el programa en ANSI C del usuario envía este mensaje a la cola de mensajes 0 ("MessageQueue 0") y cada vez que se obtiene una respuesta del CM, ésta provendrá en forma de mensaje desde la cola de mensajes 1 ("MessageQueue 1"). Estos mensajes serán despachados por el planificador de la máquina virtual ("Scheduler") de acuerdo a su orden de entrada. Los *mensajes ("messages"), colas de mensajes ("message queue"), tareas("tasks")* y el *planificador ("scheduler")* son términos comunes de la programación de los sistemas embebidos. El lector puede acceder a una introduc-

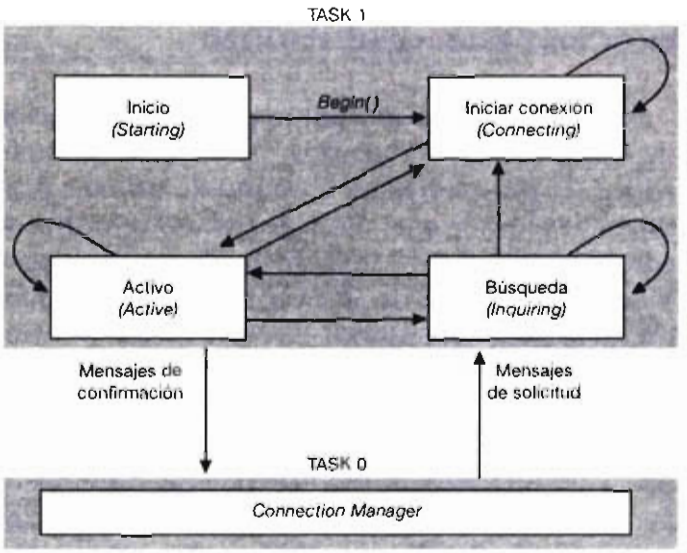


Fig. 14. Esquema de los estados del proceso de conexión de la pila BlueStack bajo el concepto de Tareas y Mensajes en el entorno de programación de un sistema embebido.

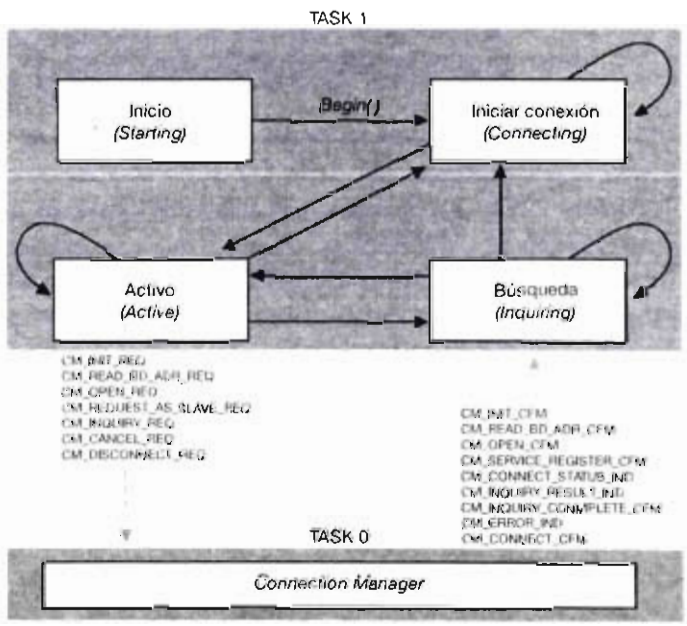


Fig. 15. Protocolo de mensajes entre el CM y la aplicación para lograr la conexión Bluetooth.

ción sobre el tema en [16] y a una discusión más profunda en [18] para poder comprender la implementación del proceso de conexión realizada en el BlueCore del CASIRA.

3.4 Programación del módulo BlueCore2 External BC2MOD2D

Se utilizó el lenguaje común de la programación de los sistemas embebidos: ANSI C. Cualquier editor de programas en C se puede utilizar para capturar el programa. En este trabajo se utilizó el editor de Visual C++ 2005 incluido en el paquete Visual Studio 2005 de Microsoft. La Fig. 16 muestra un aspecto del editor.

La programación del módulo es complicada porque requiere de un buen conocimiento de la utilidad de cada función o librería. La Fig. 17 muestra la organización de las librerías del BlueLab de las que se puede disponer para implementar una aplicación particular.

La metodología empleada para la confección del programa es la recomendada por los desarrolladores del BlueLab:

a) Distinguir y comprender los propósitos de cada uno de los tipos de librerías definidos: librerías básicas, librería CSR y librerías de aplicación (ver Fig. 17).

b) Poseer buenas nociones de la programación en lenguaje C y de nociones básicas de la programación C para sistemas embebidos.

c) Cargar, compilar, grabar y probar los programas de ejemplo que se incluyen con el BlueLab.

d) Confeccionar el programa de aplicación basándose en las aplicaciones de ejemplo más cercanas a la aplicación del usuario.

e) Solicitar asesoría técnica a CSR.

En particular, para la aplicación requerida por la plataforma de desarrollo de sensores ITS inalámbricos, el programa empleado para el módulo

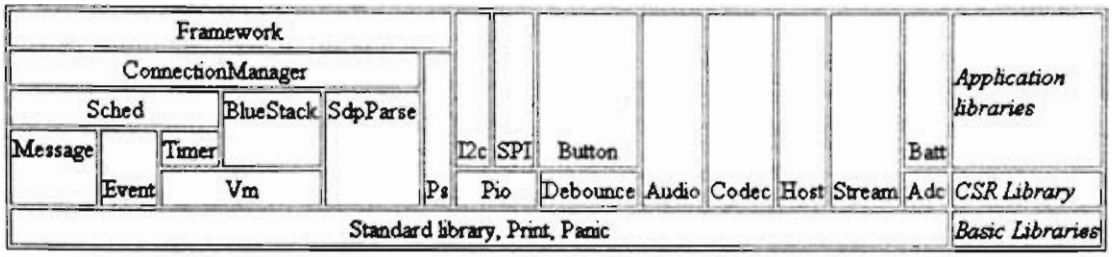


Fig. 17. Esquema de la organización de las librerías del BlueLab 2.82.

CM_CONNECT_CFM

Payload Field	Description
connect_status_t status	result of connection attempt
BD_ADDR_T addr	The address of the other device
uint16 rfc_frame_size	Max allowed length for data packets to be transmitted.
Source source	Source buffer containing the incoming data on this connection.
Sink sink	Sink buffer for writing outgoing data on this connection.
uint16 conn_server_chan	Local RFCOMM server channel that the connection was created on.

Fig. 18. Tabla que muestra la organización de la primitiva CM_CONNECT_CFM. Tomado de [19].

BlueCore2-External se basó en la aplicación llamada `spp_slave` que normalmente se incluye con los archivos del SDK. Este programa implementa un "reemplazo de cable serie" que tiene el propósito de crear un enlace RFCOMM del tipo 2 y donde el módulo se conecta como esclavo. El programa se modificó[9] de varias maneras:

a) El procedimiento de "reconocimiento" ("pairing") se omitió porque no se necesitaba de una comunicación codificada (la información del sensor no es crítica).

b) Por lo tanto, no se requería de una solicitud de clave ("PIN_CODE") o de una solicitud de llave de enlace ("LINK_KEY").

c) Como el perfil de puerto serie es inherente al uso de la capa RFCOMM, es irrelevante confirmar un registro de servicio ("SERVICE_REGISTER_CFM").

d) La definición de características como el "Parking" y el "Sniffing" no son necesarias porque la plataforma no se diseña para ser portátil.

Aquí es donde se constata la flexibilidad que ofrece la herramienta CASIRA al usuario, ya que ésta le permite determinar las características del enlace de acuerdo a su aplicación. A continuación se presenta, con más detalle, la implementación de la transmisión de datos provenientes del sensor o su emulador.

La primitiva `CM_CONNECT_CFM` es una primitiva (mensaje) que el CM regresa a la aplicación para indicar el resultado del intento de conexión. Esta primitiva se implementa como una *estructura* en lenguaje C que en sus campos guarda la información relacionada con el intento de conexión. La Fig. 18[10] muestra el concepto de la información contenida en la primitiva.

9 Consultar [15] para comprender los términos y las características modificadas en la implementación del programa del módulo BlueCore.

10 Se han respetado los nombres en inglés porque así se manejan en el programa.

Breve descripción de los campos de la primitiva

La estructura de la primitiva está definida en el archivo cm_rfcomm.h. Los campos de la primitiva son:

1) connect_status_t status. En este campo se guarda el resultado principal del intento de conexión: `CmConnectComplete`, `CmConnectTimeout`, `CmConnectCancelled`, `CmConnectDisconnect`, `CmConnectDisconnectAbnormal`, `CmConnectRemoteRefusal`, `CmConnectServiceNotSupported`, `CmConnectFailed`. Cuando la "Task 1" está en el estado activo, el mensaje que se obtiene en este campo es "`CmConnectComplete`" con lo cual es posible comenzar a enviar (o recibir) datos.

2) BD_ADDR_T addr. En este campo se tiene la dirección del dispositivo remoto con el cual se ha establecido la comunicación.

3) uint16 rfc_frame_size. En este campo se tiene la máxima longitud de los paquetes de datos que podrán ser transmitidos por el dispositivo local. Este dato no se puede modificar por el usuario.

4) Source source. Es un apuntador tipo "source" (fuente), de definición particular en las librerías del BlueLab, que apunta a una región de memoria ("buffer") donde se pueden recuperar datos de la conexión sobre el RFCOMM. Este apuntador es útil para poder manipular los datos recibidos por el dispositivo Bluetooth local.

5) Sink sink. Es un apuntador tipo "sink" (pozo), de definición particular en las librerías del BlueLab, que apunta a una región de memoria ("buffer") desde la cual se pueden enviar datos sobre el RFCOMM. Este apuntador es útil para poder manipular los datos que se desea enviar al dispositivo Bluetooth remoto.

6) uint16 conn_server_chan. Es el número del canal RFCOMM que la conexión ha creado en el dispositivo local ("server"). Este parámetro es relevante cuando un servidor debe multiplexar la comunicación entre varios esclavos en una piconet.

7) Para comprobar si la conexión ya se encuentra en el estado ACTIVO (ver Fig. 15), la aplicación verifica si en el campo *connect_status_t status* se ha colocado la sentencia ("string") "`CmConnectComplete`", con lo cual se puede iniciar el envio y/o recepción de datos a través del canal RFCOMM. Vale la pena describir la operación de recepción/transmisión de datos. La siguiente porción del código del programa del Apéndice A realiza la verificación del estado activo del proceso de conexión:

```
// Establece el canal RFCOMM (si es posible)
// 1.-          case CM_CONNECT_CFM:
             {
// 2.-                    CM_CONNECT_CFM_T *prim = (CM_CONNECT_CFM_T *) msg;
// 3.-                    connect_status_t result = prim->status;
// 4.-                    if(result == CmConnectComplete)
                    {
// 5.-                         rfcomm_source = prim->source;
// 6.-                         rfcomm_sink = prim->sink;
// 7.-                         far_addr = prim->addr;
// 8.-                         write_far_addr();
// 9.-                         status("connect_cfm complete");
//Avisa que la máquina de estados está en estado "activo"
// 10.-                   state = active;
// 11.-                   PioSet(LED_CONNECT, ~0);
                              .
                              .
                              .
```

Explicación

Línea 1. Antes de establecer el estado ACTIVO, el CM envia el mensaje CM_CONNECT_CFM que contiene la información sobre el intento de conexión. La aplicación despacha los disversos mensajes enviados por el CM a través de una declaración "case" que da la respuesta particular a cada mensaje enviado.

Línea 2. Se declara un apuntador a los campos de la estructura que representa el mensaje CM_CONNECT_CFM.

Línea 3. Se declara la variable `result` para que el apuntador le pase el valor de status.

Línea 4. Se verifica si la conexión está lista. Si el resultado de la conexión es positivo (la condición `CmConnectComplete` es verdadera), entonces:

Línea 5. Se recupera el apuntador del "buffer" de entrada en `rfcomm_source`.

Línea 6. Se recupera el apuntador del "buffer" de salida en `rfcomm_sink`.

Línea 7. Se recupera la dirección del dispositivo remoto y

Línea 8. Se guarda.

Línea 9. La aplicación anuncia que se ha completado la conexión.

Línea 10. La variable de estados cambia a `active`.

Línea 11. Finalmente, se enciende el LED 07 del CASIRA para anunciar una conexión exitosa con un dispositivo Bluetooth remoto utilizando una función de la librería PIO del BlueLab.

A partir de aquí, es posible enviar datos a través del canal RFCOMM establecido. La transmisión y/o recepción de datos es posible gracias al uso de las funciones de la librería STREAM del BlueLab. Se presenta la porción del programa que realiza la transmisión de datos junto con una explicación de su operación.

```
// 1.-          uart = StreamUartSource();
// 2.-while(1)
    {
// Polling al puerto serie (salida del sensor)
// 3.-          for(;;)
                {
// 4.-                     if(SourceSize(uart) >= LIM) break;
// 5.-                     VmWait(20);
                }
// Si el espacio reclamado está disponible, se envian los datos a través del
    canal RFCOMM

// 6.-          if(SinkClaim(rfcomm_sink, LIM) != 0xFFFF)
                {
// 7.-                     memcpy(SinkMap(rfcomm_sink), SourceMap(uart), LIM);
// 8.-                     if(SinkFlush(rfcomm_sink, LIM) == 0)
                        {
                              printf("Falla\n");
                        }
// 9.-                     SourceEmpty(uart);     //Limpia el buffer de la uart
    }
```

Explicación.

Línea 1. `uart` es un apuntador de tipo "source" declarado previamente y que, por comodidad, va a substituir al apuntador que retorna la función StreamUartSource() (librería STREAM[11]) que apunta al "buffer" que recibe los datos desde el puerto serie RS-232 del sensor o su emulador.

Línea 2. Mientras la conexión esté abierta, se transmitirán los datos del sensor hacia el dispositivo Bluetooth remoto.

Línea 3. Se interroga al puerto serie (salida del sensor) para saber si hay datos listos para transmitir. Si el número de bytes es de un tamaño mayor o igual a cierto límite (`LIM`), entonces:

Línea 4. se realizará la transmisión, de lo contrario, se continuará interrogando al puerto serie introduciendo antes una pausa de 20 m[s] (línea 5) que es posible generar con las funciones de la librería de la máquina virtual del BlueLab. Esta pausa es necesaria porque la UART del sensor se programa para transmitir a 115 200 bps, la cual es una velocidad lenta para el ritmo de procesamiento del BlueCore2. El valor de `LIM` depende de las necesidades del usuario, pero en general es de valor pequeño (<10) para no reclamar una cantidad de memoria excesiva en el BlueCore al momento de enviar este paquete de información.

Línea 6. Se reclama un espacio en memoria igual a `LIM` para el "buffer" de los datos de salida. Si la función `SinkClaim()` retorna un valor igual a `0xFFFF`, entonces no habrá espacio en la memoria RAM del BlueCore2 para crear el "buffer" de los datos a transmitir.

Línea 7. Se crea el "buffer". Se trata de apuntar al inicio de una región de memoria disponible `SinkMap()` para iniciar ahí la copia de los datos del sensor, direccionados por `SourceMap(uart)`, y dejar un apuntador `rfcomm_sink` que indique a la función de transmisión donde está el inicio del "buffer" con los datos a transmitir. Se copia un número de bytes igual a `LIM`.

Línea 8. Si la condición para transmitir los datos es válida (la función `SinkFlush()` no debe retornar un valor igual a 0), entonces se transmite el número de bytes igual a `LIM` que residen en el "buffer" cuyo inicio está apuntado por `rfcomm_sink`.

Línea 9. Finalmente, antes de regresar a interrogar al puerto serie por más datos a transmitir, la función `SourceEmpty(uart)` limpia el "buffer" de entrada y coloca al apuntador `uart` al inicio de éste.

Mientras la conexión se sostenga, el BlueCore2 enviará los datos que aparecen en el puerto serie RS-232 del CASIRA en un número de paquetes igual al valor de LIM hacia el dispositivo Bluetooth remoto.

En el Apéndice B se reporta el listado del programa "Makefile" requerido para la compilación del programa en lenguaje C del módulo BlueCore02. El programa principal de la aplicación (main.c), los programas de cabecera (service_record.h, etc.) y el Makefile deben residir en la misma carpeta para realizar correctamente la compilación.

Una vez que se cumple esto, se invoca al compilador picando en el icono  del escritorio. Recuérdese que la versión del compilador utilizado funciona sólo para un entorno Windows 2000 y será necesario tener una partición adicional con este entorno en la PC de trabajo. La Fig. 19 muestra el aspecto de la línea de comandos del compilador.

- Para iniciar la compilación, se introduce el comando >make. Si la compilación se realiza correctamente, ningún mensaje de error se generará. En caso contrario, se deberán hacer las correcciones en el programa principal y recompilar.
- Cuando se ha completado la compilación, se puede grabar la "imagen" del programa C en la EPROM del BlueCore02 (en realidad se graba el archivo HEX generado durante la compilación). Para esto, se introduce el comando >make bc02, con lo cual se realiza la grabación a la EPROM. Evidentemente, el CASIRA debe estar alimentado y el cable de comunicación del

11 Se recomienda al lector referirse a [20] para una explicación de la operación de las funciones de la librería "Stream" contenida en el BlueLab.

programa (cable paralelo conectado al puerto LPT1 de la PC y al conector CN16 del CASIRA) ha de estar presente. La disposición del cable y la configuración del CASIRA para aceptar la programación desde el puerto paralelo de la PC se documenta en [21] y [22, p. 15], respectivamente. Si la operación de grabado ha sido exitosa, el mensaje "Running code!" se visualizará. El comportamiento programado del CASIRA debe hacerse inmediatamente evidente.

3.5 Método para probar la comunicación

A fin de probar la comunicación entre el CASIRA y otro dispositivo Bluetooth, se completó la plataforma como se describe en la sección 3.1 y como se muestra en la Fig. 2.

El software de instalación del dispositivo Bluetooth de la casa Zaapa facilita su uso bajo el entorno Windows XP y la definición de los puertos serie asociados al canal RFCOMM de este dispositivo. La GUI del usuario permite realizar una búsqueda ("inquiring") de los dispositivos en la vecindad del dispositivo y elegir aquel con el cual se desea establecer comunicación.

Establecimiento de la comunicación

Los pasos a seguir para realizar la comunicación son:

1) Encender el CASIRA y verificar que el módulo se encuentra en algún estado diferente al ACTIVO (se puede verificar encendiendo un LED sobre el PCB del CASIRA).

2) Arrancar la GUI de usuario del dispositivo remoto comercial. Elegir el dispositivo al cual se quiere conectar (el módulo BlueCore02 aparecerá como CSR-bc2) y elegir la opción "Connect" en el menú de usuario de la GUI. De esta manera, el dispositivo remoto actuará como maestro al solicitar conectarse al módulo descubierto por él.

3) Esperar y verificar que se ha realizado la conexión. El LED 07 del CASIRA debe encenderse para indicar que el dispositivo se encuentra en el estado ACTIVO.

La Fig. 20 muestra el aspecto de la GUI una vez que los dispositivos BLuetooth se han conectado.

Simulación de la salida de un sensor.

A fin de probar el envio de datos a través del módulo BlueCore02, se modificó un programa LabView [23] de envio de datos a través de un puerto serie. La Fig. 21 muestra el diagrama a bloques del vi correspondiente y enseguida se explica su operación.

Fig. 19. Imagen de la línea de comandos del compilador gcc.

Fig. 20. Aspecto de la GUI del dispositivo Bluetooth comercial cuando ya se ha conectado al módulo BlueCore02.

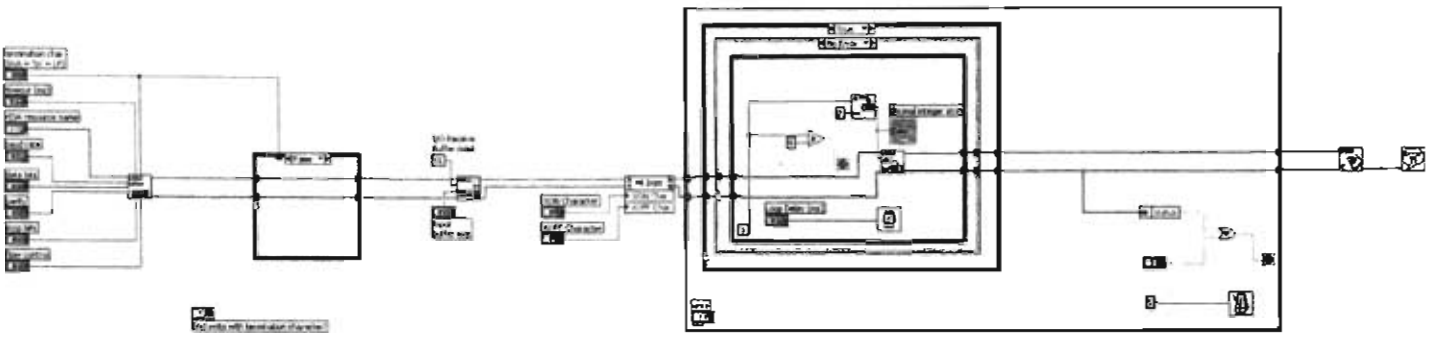


Fig. 21. Aspecto del diagrama a bloques del programa vi para visualizar los datos enviados por el módulo BlueCore02.

Explicación.

• El icono  se configura para que los parámetros de operación del puerto serie asociado con el canal RFCOMM del dispositivo local (slave) coincidan con los del puerto serie del CASIRA. En este caso se eligió la configuración: 115200-8-n-1. el control de flujo es opcional, aunque no se requiere porque no se tiene una conexión DCE a priori. No se elige ninguna configuración para el control de flujo ("none").

• El caracter de terminación también es opcional. Si el sensor está continuamente enviando datos, no es relevante definir un caracter que termine la transmisión. A veces, se necesita algún control de la transmisión del sensor (i.e. analizar un rango de muestras, reiniciar la comunicación en un periodo posterior, suspender la operación ante una anomalía, etc.), en ese caso se podrá elegir un control de flujo por software.

• Es irrelevante elegir los caracteres de control del flujo si se ha elegido dejar esta opción sin control.

• Para facilitar la visualización posterior de los datos, se incluyó un bucle que permite escribir al puerto serie  una serie continua de números en el rango 0-10 con un cierto retardo. El retardo es necesario para asegurar que el número se ha escrito correctamente al puerto. El bucle que reinicia la escritura de la serie de números también incluye un retardo que puede programarse al mínimo valor.

• Cuando se termina la operación de escritura, se cierra el puerto para que la PC disponga de él de otra forma.

La Fig. 22 muestra el aspecto del panel frontal del programa vi para simular la salida del sensor.

En general, el COM1 siempre está disponible como puerto serie RS-232 de trabajo en una PC. Los valores idóneos del retardo en la generación de la serie numérica ("Loop Delay") y del retardo en la espera de la transmisión ("timeout") se reportan al final de esta sección.

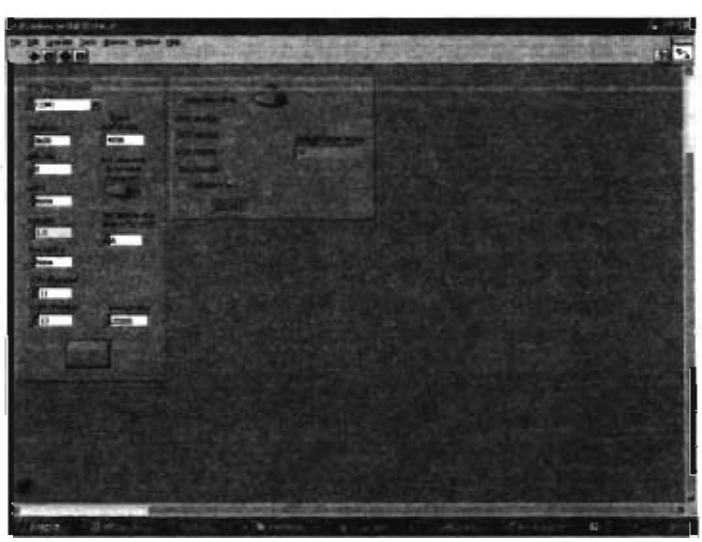


Fig. 22. Aspecto del panel frontal del programa vi para simular la salida de un sensor.

Recepción de los datos

La prueba de recepción de datos se hace en la PC a la que se le ha enchufado el dispositivo Bluetooth remoto comercial. La Fig. 23 muestra el concepto del conjunto de herramientas utilizadas para la simulación y prueba de la plataforma.

Entonces, sobre la misma PC se pone en marcha el panel frontal para simular la salida del sensor y la herramienta para visualizar los datos enviados por el CASIRA. En este último caso, se emplea la Hiperterminal de Windows conectada al puerto COM40 (definido por el software del dispositivo Bluetooth) y con los mismos parámetros de operación que los del puerto serie del CASIRA.La hiperterminal se ha de configurar para visualizar los datos enviados en diferentes lineas. La Fig. 24 muestra el aspecto de la hiperterminal mientras recibe los datos provenientes del CASIRA.

Una PC poderosa podría permitir el uso simultáneo de dos sesiones de LabView que se ocupen de la simulación del sensor y de la recepción de datos. De otra manera, una de las dos sesiones se atascará o la operación simultánea será lentísima y se perderá información.

En este caso, se conectó la hiperterminal al puerto COM40 y, enseguida, se realizó la conexión RFCOMM desde la GUI de usuario del dispositivo Bluetooth cmercial (opción "connect").

NOTA IMPORTANTE. Se ha de conectar la hiperteminal al puerto COM asignado antes de establecer la conexión RFCOMM. De otra manera, la recepción de información no será continua y la operación se detendrá a los pocos segundos de haber inciado. auqnue la razón de esto no es clara.

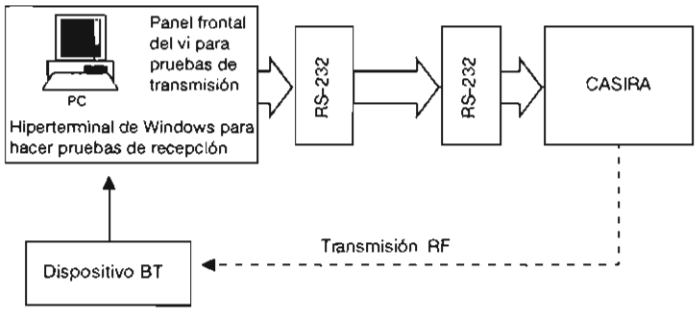


Fig. 23. Diagrama a bloques de las herramientas de prueba para la plataforma.

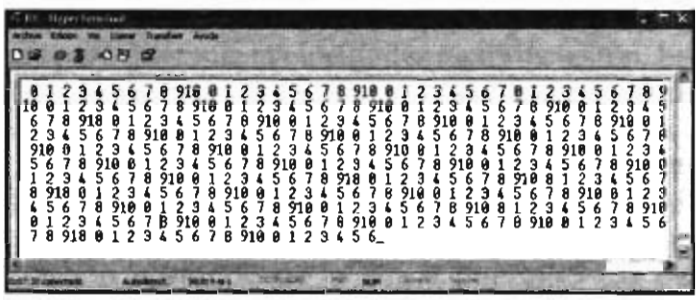


Fig. 24. Aspecto de la hiperterminal donde se visualizan los datos enviados por el CASIRA.

4. Resultados correspondientes a la plataforma

La configuración de la UART del BlueCore02 puede cambiarse con la siguiente línea en el programa principal del módulo:

```
StreamUartConfigure(VM_UART_
RATE_9K6, VM_UART_STOP_ONE, VM_UART_
PARITY_NONE);
```

Que ha de colocarse antes de que el planificador ("scheduler") de la máquina virtual sea lanzado. Sin embargo, esta línea no tendrá efecto si la UART no se habilita desde el archivo Makefile (el archivo que ayuda a crear el archivo objeto durante la compilación). Por lo tanto, en esta aplicación es indispensable el comando:

```
TRANSPORT = raw
```

en el archivo Makefile (cf. Apéndice B). El Puerto RS-232 del CASIRA se programó para transmitir 8 bits a 115200 baudios, sin paridad y con un bit de "stop" (115200-8-n-1). La Tabla III muestra el retardo mínimo en la generación de datos ("loop delay") manejada por el vi para simular la salida del sensor y el mínimo tiempo de espera de transmisión del puerto ("timeout") antes de que algún dato (byte) se pierda en la recepción. Esto se muestra para diferentes velocidades de transmisión. Estos valores asegurarían la máxima proporción de escritura al puerto sin pérdida de información.

De acuerdo a los datos de la Tabla III, si once bytes se escriben cada 18 m[s] (el "loop delay" programado en el vi), un máximo de 5000 bps se estarían enviando de la PC al puertoRS-232 del CASIRA sin perder los datos monitoreados sobre la hiperterminal. En las conclusiones se discute esta figura.

Tabla III. Caracteristicas de transmisión del radio Bluetooth implementado.

Velocidad [bps]	*Loop Delay m[s]*	*Timeout m[s]*
9600	*18*	*> 600*
38400	*18*	*> 600*
115200	*18*	*> 600*

5. Uso de la plataforma de comunicación para un sensor de espira inductiva

Una vez ensayada la transmisión de datos de forma serie con el radio Bluetooth del CASIRA, se procedió a conectar uno de los detectores inductivos de espira desarrollados por el Grupo de Sistemas de Control de Tráfico (SCT) adscrito al ITACA-UPV. Este detector genera una señal útil para la clasificación de automóviles o medición de la velocidad de los autos [24]. El detector y su procesador de señal forman un sensor inteligente capaz de calcular datos en tiempo real. La transmisión de datos en tiempo real desde el sensor inteligente, el registro de los datos procesados del sensor, el análisis posterior de los datos procesados por el sensor y las pruebas al sensor, para fines de mantenimiento, son tareas que pueden llevarse a cabo a través de un sistema de comunicación inalámbrica que evitaría la instalación de un enlace físico entre el sensor inteligente y el equipo electrónico especial colocado a lo largo de la vía. La Fig. 25 muestra la infraestructura relacionada con el detector de espira inductiva aplicada al control del tráfico.

En esta figura, se aprecia que la comunicación entre la unidad electrónica en el punto A con la estación de concentración de datos en el punto B sería problemática porque habría que ocultar el cable bajo el camino o construir una infraestructura nueva, robusta y cara para instalar el cable de un lado del camino al otro. Es evidente que un sistema de comunicación inalámbrica facilita la tarea de la comunicación entre los sensores y la infraestructura a lo largo del camino.

El detector de espira inductiva tiene un DSP como órgano de cálculo principal equipado con una salida RS-232. Esta salida es el caso común en muchos sensores ITS (sensores de temperatura, humedad, niebla, velocidad, distancia, etc.). El DSP forma una serie de 30 bits en 230 μs y lo envia a través del puerto serie en un intervalo de 270 μs, lo que significa tener una velocidad de transferencia

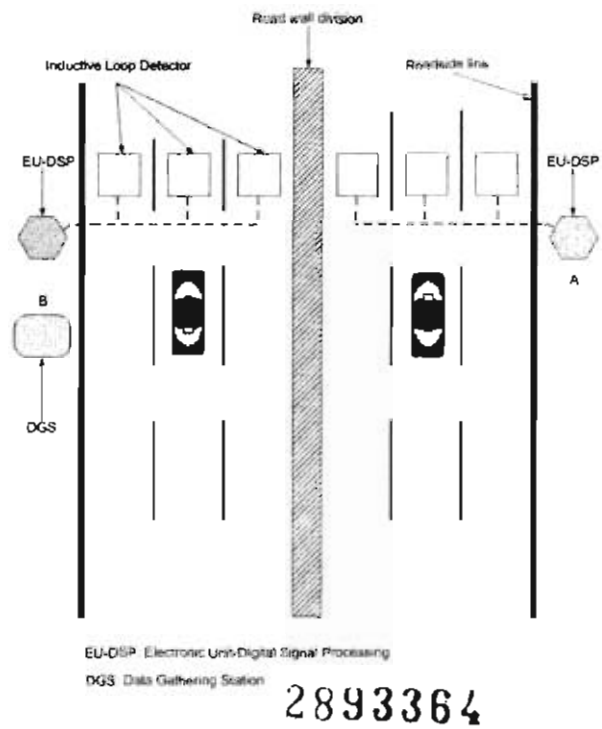


Fig. 25. Esquema que muestra el uso de los detectors de espira infductiva y la infraestructura asociada a lo largo de la autovía.

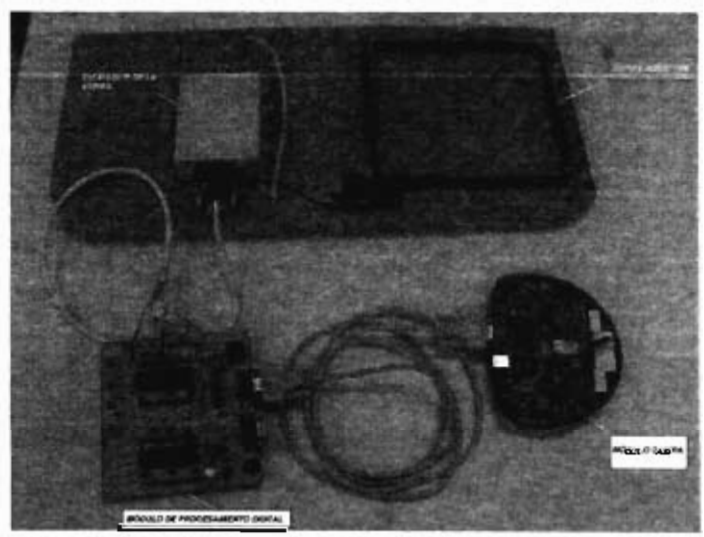
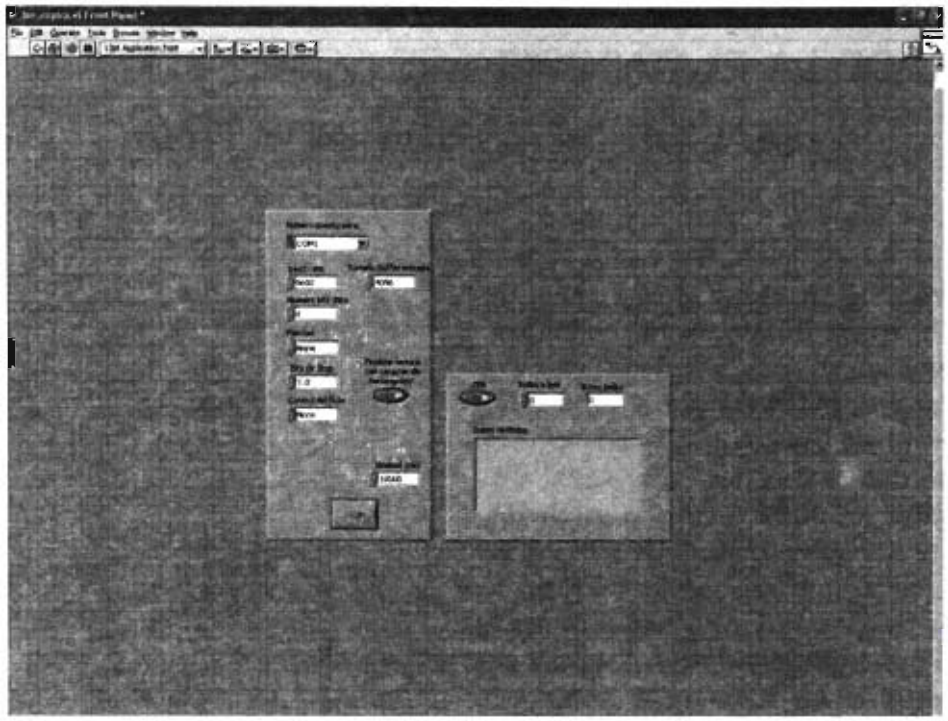
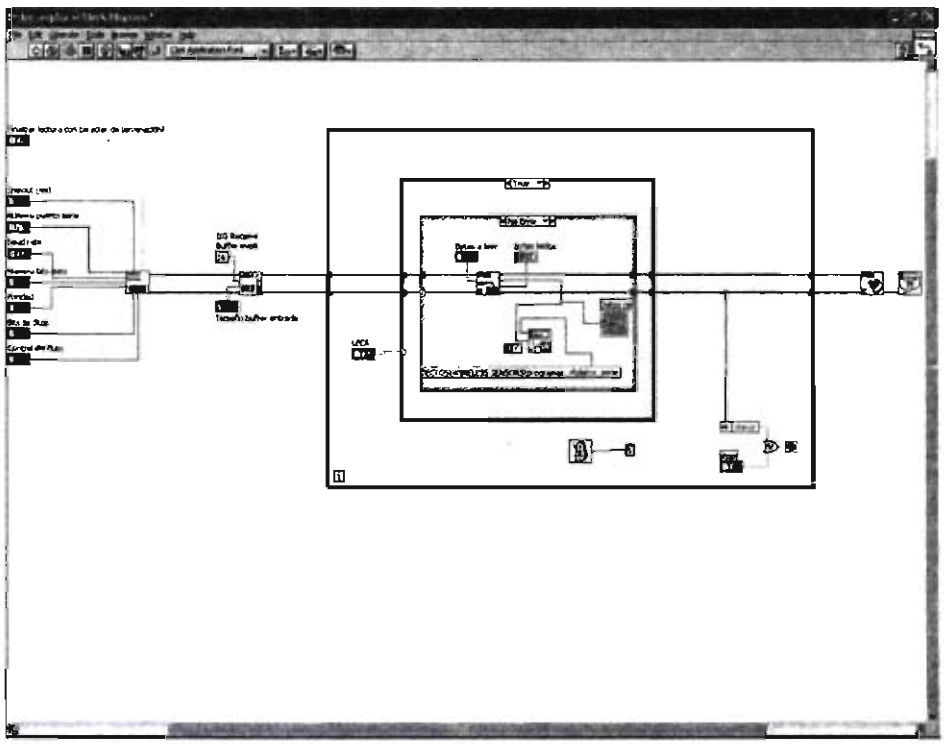


Fig. 26. Fotografía del sistema de control de flujo provisto del módulo CASIRA para comunicación inalámbrica.



a)



b)

Fig. 27. a) Panel frontal del programa vi para la lectura de los datos enviados por el detector usando comunicación Bluetooth. b) Diagrama a bloques del programa vi de recepción de datos.

de 111 111 bps muy cercana a la velocidad de 115 200 bps de un puerto RS-232. Como ya se apuntó en la sección 2, un radio Bluetooth es capaz de manejar estas velocidades de transferencia cuando se emula un puerto RS-232 mediante la capa RFCOMM.

5.1 Solución material de la comunicación inalámbrica del sensor de espira inductiva

En la Fig. 26 se muestra una fotografía del prototipo del detector de espira inductiva provisto del módulo Bluetooth para la comunicación inalámbrica de datos. Actualmente, este detector se utiliza para clasificar automóviles.

Como lo muestra la Fig. 26, el detector está compuesto por la espira inductiva, su excitador y la unidad de procesamiento digital. Una descripción más detallada de este sistema se hace en [24]. La lógica del detector está programada para generar un número de identificación de la espira y dos lecturas correspondientes a cambios en el voltaje que representa el campo magnético de la espira cuando un automóvil pasa sobre ella. Esta es la información que se transmite a la unidad concentradora de datos.

5.2 Resultados de la transmisión con el sensor de espira

Para esta memoria, las pruebas de transmisión y recepción se realizaron en laboratorio. Los resultados obtenidos fueron satisfactorios.

Tabla IV. Muestra de los datos recibidos desde el sensor de espira inductiva.

1 3512
2 2750
3 2224
1 3516
2 2735
3 2215
1 3517
2 2737
3 2214
1 3516
2 2737
3 2214

Para la recepción de datos, se modificó nuevamente el programa vi de LabView, utilizado para las pruebas con la plataforma, para poder visualizar los datos recibidos por el "dongle" Bluetooth conectado a la PC. La Fig. 27 muestra el aspecto del panel frontal de este programa y el diagrama a bloques correspondiente.

En la tabla IV muestra los datos recibidos y almacenados en el archivo de almacenamiento preparado por el vi de la Fig. 27 c). El primer número es el número de identificación de la espira. Los cuatro números siguientes se deben leer por pares y corresponden a valores de los cambios en el campo magnético de la espira cuando es perturbado por un objeto.

6. Conclusión

Las características propias de cada una de las herramientas de desarrollo utilizadas en este trabajo (el sistema embebido del CASIRA, LabView y la Hiperterminal de Windows), han sido muy útiles en el diseño e implementación de la plataforma de desarrollo Bluetooth para sensores inalámbricos orientados a aplicaciones ITS. Con esta plataforma, es posible proveer a un sensor con salida RS-232 de la funcionalidad Bluetooth para una velocidad de transferencia máxima de 115 200 bps. Esto es aceptable para los sensores ITS de baja proporción en la generación de datos.

La plataforma entonces permitiría un rápido equipamiento de la funcionalidad Bluetooth para sensores ITS comerciales o propietarios.

Un ordenador portátil (laptop) con funcionalidad Bluetooth y trabajando como maestro en un entorno punto a punto es un sistema más conve-niente para la monitorización de sensores ITS como la espira inductiva.

Un trabajo futuro en relación con esta plataforma será su implementación en campo y el estudio de su comportamiento ante obstáculos en la autovía como grandes contenedores remolcados por trailers, condiciones de clima adversas (lluvia, nieve), la integración de un sistema completo donde se prescinda del módulo CASIRA y una comparación con equipos similares.

REFERENCIAS

[1] Jun Luo, Jean-Pierre Hubaux, A survey of Inter-Vehicle Communication, School of Computer and Communication sciences EPFL, CH-1015 Lausanne, Switzerland, Technical Report IC/2004/24.

[2] Weidong Xiang, Paul Richardson, Jinhua Guo, Introduction and Preliminary Experimental Results of Wireless Access for Vehicular Environments (WAVE) Systems, (invited paper), V2VCOM, San José, California, July 2006.

[3] http://grouper.ieee.org/groups/802/11/Reports/802.11_Timelines.htm

[4] Gregory S. Bickel, Inter/Intra-Vehicle Wireless Communication, http://www.cse.wustl.edu/~jain/cse574-06/ftp/vehicular_wireless.pdf

[5] Nick Baker, Zigbee and Bluetooth Strengths and Weaknesses for Industrial Applications, IEE Computing and Control engineering Journal, Vol. 16, Issue 2, April-May 2005, pp. 20-25.

[6] Peter R. Hauptmann, Selected Examples of Intelligent (micro) Sensor Systems: State-of-the-Art and Tendencies, Meas. Sci. Technol. 17 (2006), 459-455.

[7] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, Erdal Cayirci, A Survey on Sensor Networks, IEEE Communications Magazine, August 2002, pp. 102-114.

[8] Industrial Networks: Wireless Communications, Contact No. 20, Special Report, Chauvin Arnoux Group, 2004, http://www.chauvin-arnoux.com/Groupe/pdf_mag/cmn20.pdf

[9] Cambridge Silicon Radio, http://www.csr.com

[10] CASIRA Bluetooth Development Kit, Ref: LT02V3.0, Cambridge Silicon Radio Ltd., 2000.

[11] BlueCore 2-External Single Chip Bluetooth System, BC212015-ds-001f, Cambridge Silicon Radio, March 2003.

[12] BlueStack User Manual, C6066-UM-001 v1.6, Mezoe 2001.

[13] http://www.jc1.se/images/bluet09.htm

[14] BlueCore 2-External Data Sheet, BC212015-ds-001f.pdf, CSR, March 2003.

[15] Jennifer Bray, Charles F. Sturman, Bluetooth Connect without Cables, 2nd Ed., Prentice Hall, 2002, ISBN 0-13-066106-6.

[16] Kammer D., McNutt G., Senese B., Bray J., Bluetooth Application Developer's Guide: The Short Range Interconnect Solution, Syngress Publishing, 2002, ISBN: 1-928994-42-3.

[17] BlueLab v2.8 Software Development Kit Installation Guide, CSR, November 2003.

[18] Michael Barr, Anthony Massa, Programming Embedded Systems with C and GNU Development Tools, 2nd Ed., O'Reilly Media Inc., 2007, ISBN -10: 0-596-00983-6, ISBN-13: 978-0-569-00983-0.

[19] CM_RFCOMM Library, BlueLab Professional v2.8, CSR, Nov. 2003.

[20] Stream Library, BlueLab Professional v2.8, CSR, Nov. 2003.

[21] SPI Connector Pinout, casi-sp-008Pa, CSR, November 2003.

[22] BlueCore Casira User Guide, bc01-an-100Pc, CSR, July 2003.

[23] Advanced Serial and Read.vi. Example posted on the NI Example Finder.

[24] Arroyo A., Mocholí A., Técnicas de Clasificación y Medida de Velocidad de Vehículos Mediante Espira Única, XII Seminario Anual de Automática, Electrónica Industrial e Instrumentación (SAAEI2005), Santander, España, Resúmenes de los Trabajos, Ed. SP Universidad de Cantabria, I.S.B.N.: 84-8102-964-5, 2005, pp. 1-5.

REPORTE DE INVESTIGACIÓN

COMUNICACIÓN BLUETOOTH PARA SENSORES UTILIZADOS EN APLICACIONES DE CONTROL DE TRÁFICO

Se terminó de imprimir en el mes de marzo de 2011 en los talleres de la Sección de Impresión y Reproducción de la Universidad Autónoma Metropolitana Unidad Azcapotzalco, Av. San Pablo 180, Col. Reynosa Tamaulipas Del. Azcapotzalco, C. P. 02200, México, DF. La edición estuvo a cargo de la Oficina de Producción Editorial y Difusión de la División de Ciencias Básicas e Ingeniería con un tiraje de 40 ejemplares.

UNIVERSIDAD AUTONOMA METROPOLITANA
Casa abierta al tiempo Azcapotzalco

COOSEI
COORDINACIÓN DE SERVICIOS DE INFORMACIÓN

Formato de Papeleta de Vencimiento

El usuario se obliga a devolver este libro en la fecha señalada en el sello mas reciente

Código de barras. 2893364

FECHA DE DEVOLUCION

- Ordenar las fechas de vencimiento de manera vertical.
- Cancelar con el sello de "DEVUELTO" la fecha de vencimiento a la entrega del libro

Grupo de Sensores y Señales
Departamento de Electrónico
División de Ciencias Básicas Ingeniería
Unidad Azcapotzalco

UNIVERSIDAD
AUTONOMA
METROPOLITANA
Casa abierta al tiempo Azcapotzalco